Nazwa implementacji: Nauka języka C – pętla while **Autor:** Piotr Fiorek

Opis implementacji:

Poznanie czym są pętle i jak korzystać z pętli while i do {} while().

W programach bardzo często się zdarza, że chcemy, aby jakaś czynność wykonała się określoną ilość razy. Właśnie do tego służą pętle. Pierwszą pętlą, jaką poznamy, jest pętla „while”.

Pierwszą pętlą, jaką poznamy, jest pętla „while”. Tak samo jak wyrażenie „if” sprawdza ona prawdziwość warunku i wykonuje kod. Jednak „while” jest pętlą więc, zamiast wykonać kod ze swojego bloku i dalej wykonywać program powtarza ona wykonanie swojego bloku kodu, po każdym razie sprawdzając czy warunek nadal jest prawdziwy i jeśli tak następuje kolejne wykonanie bloku kodu, po nim kolejne sprawdzenie warunku i tak dopóki warunek nie stanie się fałszywy. Dlatego właśnie w pętlach zawsze bardzo ważne jest uważne dobieranie warunków tak, aby w końcu ich wartość zmieniła się na fałszywą i program mógł się wykonywać dalej.

Wyjątkiem od tej zasady są tak zwane pętle nieskończone. Są to pętle, w których warunek z założenia zawsze jest prawdziwy, a zakończenie wykonania pętli następuje poprzez wykorzystanie wcześniej poznanej instrukcji „break”. Powoduje ona natychmiastowe przerwanie wykonania i wyjście z bloku kodu pętli.

Schematycznie pętla „while” wygląda tak:

```
while(warunek)

{

instrukcja1;

instrukcja2;

...

}
```

Przeróbmy teraz nasz kalkulator tak, aby używając pętli, pozwalał wykonywać obliczenia dowolną ilość razy i za każdym razem na końcu pytał czy użytkownik chce wykonać kolejne działanie czy opuścić program:

```
#include <stdio.h>

int main(void)

{

int zmienna, cyfra1, cyfra2, wynik, wyjscie=0;

char kontynuj;
```


KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI
ORE
UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt “Strategia Wolnych i Otwartych Implementacji jako innowacyjny model zainteresowania kierunkami informatyczno-technicznymi oraz wspierania uczniów i uczennic w kształtowaniu kompetencji kluczowych” współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

```
while(wyjscie == 0)

{

printf("Jaką operację chcesz wykonać?\n1) dodawanie\n2) odejmowanie\n3) mnożenie\n4) dzielenie\n");

scanf("%d", &zmienna);

printf("Podaj pierwszą cyfrę do operacji: ");

scanf("%d", &cyfra1);

printf("Podaj drugą cyfrę do operacji: ");

scanf("%d", &cyfra2);

switch(zmienna)

{

case 1:

wynik = cyfra1 + cyfra2;

break;

case 2:

wynik = cyfra1 - cyfra2;

break;

case 3:

wynik = cyfra1 * cyfra2;

break;

case 4:

if(cyfra2 == 0)

{

printf("Nie wolno dzielic przez zero\n");

return 1;

}

wynik = cyfra1 / cyfra2;

break;
```


KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI
UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt “Strategia Wolnych i Otwartych Implementacji jako innowacyjny model zainteresowania kierunkami informatyczno-technicznymi oraz wspierania uczniów i uczennic w kształtowaniu kompetencji kluczowych” współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

```
default:
printf("Wybrales nieistniejaca opcje\n");
return 1;
}
printf("Wynik wynosi: %d\n", wynik);
while(1)
{
printf("\nCzy chcesz wykonac kolejne obliczenie? (t/n): ");
getchar();
scanf("%c", &kontynuj);
if(kontynuj == 't')
{
break;
}
else if(kontynuj == 'n')
{
wyjscie = 1;
break;
}
else
{
printf("Podales bledna litere!\n");
}
}
}
return 0;
}
```


KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI
ORE
UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt “Strategia Wolnych i Otwartych Implementacji jako innowacyjny model zainteresowania kierunkami informatyczno-technicznymi oraz wspierania uczniów i uczennic w kształtowaniu kompetencji kluczowych” współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

Jak widać cały dotychczasowy kod (z wyjątkiem deklaracji zmiennych, ponieważ deklarujemy je tylko raz na cały program, a nie za każdym „okrążeniem” pętli) został włączony do bloku kodu pętli, ponieważ chcemy, aby wykonywał się on wiele razy. Na samym początku widać warunek pętli. Nasza pętla będzie się wykonywać tak długo jak długo zmienna „wyjscie” jest równa zero. W momencie, kiedy zmienimy wartość zmiennej na dowolną inną, przy ponowny sprawdzeniu warunek nie będzie już prawdziwy i kod pętli nie zostanie wykonany.

Pod koniec kodu pętli umieściliśmy drugą pętlę, tym razem nieskończoną. Jej warunkiem jest po prostu cyfra jeden. Jak mówiłem wcześniej, w C fałsz ma wartość zero, a każda inna wartość jest uznawana za prawdę. W związku z tym warunek w postaci dowolnej cyfry różnej od zera zawsze jest prawdziwy. W tej pętli zadawane jest pytanie czy wykonać kolejne obliczenie. Na początku zadawane jest pytanie czy użytkownik chce wykonać kolejną operację, później korzystamy z funkcji „getchar()”, aby opróżnić bufor znakowy, co jest potrzebne, ponieważ tym razem pobieramy z klawiatury literę. Następnie znaną już funkcją „scanf” pobieramy wartość z klawiatury (tym razem używamy przełącznika „%c” i zmiennej typu „char”, ponieważ chcemy pobrać literę) i w blokach warunkowych decydujemy co zrobić dalej. Jeśli chcemy kontynuować obliczenia, po prostu przerywamy drugą pętlą, jeśli chcemy wyjść z programu, zmieniamy wartość zmiennej sterującej pierwszą pętlą i wychodzimy z drugiej, a jeśli użytkownik podał błędną literę, pozostajemy w pętli nieskończonej, wyświetlając komunikat o błędzie i zadając jeszcze raz pytanie co zrobić dalej.

Poza pętlą „while” istnieje też pętla „do {} while();”, która ogólną zasadę działania ma taką samą jak pętla „while”, ale różni się jednym zasadniczym szczegółem – w pętli „do {} while();” najpierw wykonywany jest blok kodu, a dopiero potem sprawdzany jest warunek. Powoduje to, że w przeciwieństwie do pętli „while” blok kodu zostanie zawsze wykonany co najmniej raz.

Zmieńmy zatem nasz kalkulator tak, aby zamiast pętli nieskończonej na końcu używał nowo poznanej pętli:

```
#include <stdio.h>

int main(void)

{

int zmienna, cyfra1, cyfra2, wynik, wyjscie=0;

char kontynuj;

while(wyjscie == 0)

{

printf("Jaką operację chcesz wykonać?\n1) dodawanie\n2) odejmowanie\n3) mnożenie\n4) dzielenie\n");

scanf("%d", &zmienna);

printf("Podaj pierwszą cyfrę do operacji: ");

scanf("%d", &cyfra1);

printf("Podaj drugą cyfrę do operacji: ");

scanf("%d", &cyfra2);

switch(zmienna)
```


KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI
ORE
UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt “Strategia Wolnych i Otwartych Implementacji jako innowacyjny model zainteresowania kierunkami informatyczno-technicznymi oraz wspierania uczniów i uczennic w kształtowaniu kompetencji kluczowych” współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

```
{
case 1:
wynik = cyfra1 + cyfra2;
break;
case 2:
wynik = cyfra1 - cyfra2;
break;
case 3:
wynik = cyfra1 * cyfra2;
break;
case 4:
if(cyfra2 == 0)
{
printf("Nie wolno dzielic przez zero\n");
return 1;
}
wynik = cyfra1 / cyfra2;
break;
default:
printf("Wybrales nieistniejaca opcje\n");
return 1;
}
printf("Wynik wynosi: %d\n", wynik);
do
{
printf("\nCzy chcesz wykonac kolejne obliczenie? (t/n): ");
getchar();
scanf("%c", &kontynuj);
```


KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI
UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt “Strategia Wolnych i Otwartych Implementacji jako innowacyjny model zainteresowania kierunkami informatyczno-technicznymi oraz wspierania uczniów i uczennic w kształtowaniu kompetencji kluczowych” współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

```
if(kontynuj == 'n')

{

wyjscie = 1;

}

else if(kontynuj != 't')

{

printf("Podales bledna litere!\n");

}

} while((kontynuj != 't') && (kontynuj != 'n'));

}

return 0;

}
```

W ten sposób program stał się czytelniejszy, ponieważ nie trzeba szukać w kodzie przypadku kiedy pętla skończy działanie, używając instrukcji „break”, tylko wystarczy spojrzeć na koniec bloku kodu do warunku w „while” i od razu wiadomo, jaki przypadek nas interesuje.

Zadania:

Napisz program wczytujący z klawiatury wartości, a następnie wyświetlający średnią tych wartości, niech program kończy wprowadzanie, kiedy natrafi na cyfrę 0;

Napisz ten sam program, używając pętli do-while.

Napisz program wypełniający tablicę dziesięcio-elementową kolejnymi liczbami parzystymi (porównaj ją później z podobną pętlą for).

Napisz program, który w zmiennej będzie przechowywał hasło i który po uruchomieniu będzie prosił użytkownika o podanie hasła i nie pozwoli przejść dalej, jeśli użytkownik nie poda właściwego hasła (do porównywania ciągów znaków użyj funkcji strcmp – man strcmp).

KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI
ORE
FWIOO
UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt “Strategia Wolnych i Otwartych Implementacji jako innowacyjny model zainteresowania kierunkami informatyczno-technicznymi oraz wspierania uczniów i uczennic w kształtowaniu kompetencji kluczowych” współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.